Differentially Private Graph Neural Networks for Link Prediction

Xun Ran†, Qingqing Ye†, Haibo Hu†, Xin Huang‡, Jianliang Xu‡, Jie Fu†
†The Hong Kong Polytechnic University, ‡Hong Kong Baptist University
qi-xun.ran@connect.polyu.hk *qqing.ye@polyu.edu.hk* *haibo.hu@polyu.edu.hk*
xinhuang@comp.hkbu.edu.hk *xujl@comp.hkbu.edu.hk* *jie.fu@stu.ecnu.edu.cn*

***Abstract*—Graph Neural Networks (GNNs) have proven to be highly effective in addressing the link prediction problem. However, the need for large amounts of user data to learn representations of user interactions raises concerns about data privacy. While differential privacy (DP) techniques have been widely used for node-level tasks in graphs, incorporating DP into GNNs for link prediction is challenging due to data dependency. To this end, in this work we propose a differentially private link prediction (DPLP) framework, building upon subgraph-based GNNs. DPLP includes a DP-compliant subgraph extraction module as its core component. We first propose a neighborhood subgraph extraction method, and carefully analyze its data dependency level. To reduce this dependency, we optimize DPLP by integrating a novel path subgraph extraction method, which alleviates the utility loss in GNNs by reducing the noise sensitivity. Theoretical analysis demonstrates that our approaches achieve a good balance between privacy protection and prediction accuracy, even when using GNNs with few layers. We extensively evaluate our approaches on benchmark datasets and show that they can learn accurate privacy-preserving GNNs and outperforms the existing methods for link prediction.**

***Index Terms*—Data privacy, Link prediction, Graph neural networks, Differential privacy**

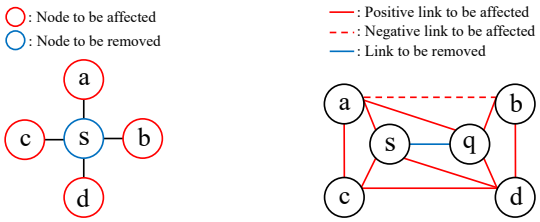
I. Introduction

Link prediction is the problem of predicting the existence of a link between two nodes within a graph [1]. Given the ubiquitous existence of graph data, this field finds applications in a multitude of domains, such as friend recommendation within social networks [2], movie recommendation on Netflix [3], co-authorship prediction in citation networks [4]. Recently, graph neural networks (GNNs) have shown superior performance in link prediction when compared to traditional methods [5], [6]. In particular, by leveraging a localized subgraph surrounding each target link, subgraph-based GNNs (SGNNs) have been demonstrated to learn the most expressive structural representations of links for prediction.

On the other side of the coin of using GNN models for link prediction, the users' privacy is at risk. Privacy breaches can happen in various ways since an attacker can infer sensitive node features or links from the trained models [7], [8]. Differential Privacy (DP) has received significant attention in incorporating it into GNN-based methods to mitigate the privacy risks. However, current studies primarily focus on developing differentially private GNNs for node classification [9]–[13], leaving a noticeable gap in research concerning link prediction. To bridge the gap, our work focuses on learning GNNs for accurate link prediction while preserving differential privacy.

Extending approaches designed for node-level tasks to link prediction is a challenging task. In GNNs, learning the representation of each node relies not only on its own features but also on aggregated information from its neighborhood. This introduces a complex data dependency issue, which poses additional challenges in the context of link prediction compared to node-level tasks. To illustrate the challenges, Figure 1 shows an example involving a 1-hop neighborhood subgraph.

Example 1. *In node-level tasks, to determine the nodes affected when a specific node s is removed, we can simply count its 1-hop neighbors (i.e., nodes a, b, c, d in Figure 1 (a)). The number of affected nodes is bounded by the maximum node degree θ, which is 4 in this case. Regarding link prediction, removing the link (s, q) in Figure 1 (b) will impact all the red links since they all have their 1-hop neighborhood subgraphs containing the link (s, q). However, enumerating these affected links becomes challenging because their relative positions to (s, q) vary from case to case (e.g., links $(q, b), (q, a), (c, d)$). Additionally, to train a link prediction model, we should also consider the non-existent links (e.g., (a, b)) sampled as negative examples. In the worst case, the total number of affected links can reach $\Omega(\theta^2)$.*



(a) Affected nodes in node-level tasks (b) Affected links in link prediction

Fig. 1: Data dependency issues in 1-hop subgraph, with the maximum node degree $\theta = 4$

Accurately measuring data dependency is crucial for calculating noise sensitivity to comply with DP. In the literature, two paradigms have been studied to address the issue of data dependency in GNNs. **The first approach is to avoid the complexity of measuring dependencies** by introducing DP

noise to either the input, such as adjacency matrix [9], or the aggregation functions [10] used in GNNs. However, the former often introduces excessive noise that compromises the model utility, and the later lacks adaptability to diverse GNN architectures since the aggregation function is typically non-learnable. Moreover, this approach suffers from a loss of model utility as the depth of GNNs increases.

The second paradigm measures the dependency among nodes, focusing on the task of node classification [13]. In particular, an upper bound of interdependent nodes is derived by sampling the incoming neighbors of any nodes in a graph. However, this approach cannot be directly applied to link prediction for several reasons: (i) The dependency analysis of node-level tasks considers neighboring nodes, but there is no off-the-shelf definition of neighboring links. (ii) Link prediction requires considering the neighborhoods of both two nodes, introducing a higher level of dependency compared to node classification. (iii) To train a link prediction model, it is essential to account for non-existent links (e.g., link (a, b) in Figure 1(b)), as their presence increases the noise sensitivity. These factors collectively contribute to the increased complexity of dependency analysis for link prediction.

In this work, we propose a differentially private link prediction (DPLP) framework based on GNNs, which is compatible with various GNN blocks with learnable aggregation functions. In the framework, the sensitivity analysis module introduces the notion of link-based subgraph contextual neighbor and gives the theoretical results of dependency level. To constrain the dependency level, our framework employs a graph projection step to bound the node degree and a subgraph extraction step with a tunable negative sampling rate. Our method yields a tight upper bound of dependency level for determining a sound noise scale to comply with DP.

To improve the utility, we propose an optimized path-based subgraph extraction module for the DPLP framework. This method is built upon the observation that the prevailing subgraph extraction technique often generates subgraphs containing dangling edges. Such dangling edges amplify inter-subgraph dependencies but offer limited contribution to link prediction. By dropping these edges, the information carried by the subgraph is confined to the paths connecting the nodes of a target link. Consequently, this reduces the dependency level while preserving link structural information. As a result, this optimization leads to higher prediction accuracy while maintaining strict privacy requirements. Overall, the contributions of our work are summarized as follows:

- We propose a differential privacy-preserving link prediction (DPLP) framework. To the best of our knowledge, this is the first framework to learn differentially private GNN for link prediction. This framework is adaptable to various GNN architectures.
- We design an optimized path-based subgraph extraction module for the DPLP framework, which reduces data dependency by focusing on the paths connecting the nodes of a target link, resulting in improved prediction accuracy.
- We conduct extensive theoretical analysis on both privacy and utility aspects. In particular, we prove that our algorithm approximates the predictive performance of multi-layer GNNs using fewer layers.
- We evaluate the effectiveness of DPLP over real-world datasets. The experimental results validate our theoretical findings and demonstrate the effectiveness of the proposed methods. The results also present that our method strikes a balance between link prediction accuracy and privacy protection.

The remainder of this paper is organized as follows. Section II reviews related works. Section III introduces preliminaries and problems on the differentially private GNN for link prediction. Section IV presents the overview and the implementation details of the DPLP framework. Section V proposes the optimized path-based subgraph extraction method. Section VI shows the theoretical analysis of the proposed methods. Section VII presents the experimental results, followed by a conclusion in Section VIII. The detailed proof of the theoretical results are in [14].

II. Related Work

Traditional link prediction techniques include heuristic, latent-feature, and content-based methods. Heuristic methods like Common Neighbors and Adamic-Adar [1], [2] estimate link probabilities based on node similarity. Latent-feature methods learn node embeddings through matrix factorization, exemplified by DeepWalk and node2vec [15], [16]. Content-based methods, on the other hand, utilize node attributes [17].

Recent advancements in link prediction are led by Graph neural networks (GNNs), which outshine traditional methods by integrating graph structure with node/edge features. Among GNN-based approaches, subgraph-based methods have emerged as particularly effective [6]. These methods focus on local subgraphs around target links, applying a GNN to learn representations for each subgraph, which then serve as link representations for prediction.

Zhang et al.'s analysis [5] reveals that subgraph-based methods excel over node-based methods in link prediction by effectively capturing the relational context within local subgraphs. While node-based methods falter in recognizing the relative positioning of node pairs due to their isolated learning approach, subgraph-based methods overcome this limitation without deep GNNs. The impracticality of deep GNNs in privacy-sensitive environments, owing to escalating data dependencies, further underscores the suitability of subgraph-based approaches, as a foundational framework for privacy-preserving link prediction.

Research in link prediction privacy is bifurcated into centralized and decentralized approaches. Decentralized scenarios involve collaboration among entities holding different parts of graph data, aiming to secure data transactions and prevent privacy breaches [9]. Centralized models, conversely, focus on developing robust privacy-preserving models to resist attacks like model inversion and membership inference [7], [8]. Significantly, research has shown that link existence alone can

disclose sensitive user information [18], [19], underscoring the importance of link-level privacy in centralized settings. This work considers the problem of link prediction preserving link-level privacy in a centralized setting.

Differential Privacy (DP) has been extensively applied to link prediction methods, including heuristic and latent-feature approaches [20]–[24]. Traditional DP applications, such as DP-SGD [25], face challenges with GNNs due to data dependency issues, where computations rely on interconnected user data, increasing privacy risks.

To address this, DP in GNNs is approached through input perturbation, aggregation function perturbation, and gradient perturbation [9], [10], [12], [13]. Input perturbation adds noise to the graph data, aggregation function perturbation modifies GNN's message passing, and gradient perturbation involves noise addition in GNN gradients. While each technique manages data dependencies, they also face limitations like signal loss or reduced learnability.

Mueller et al. proposed a DP method for graph-level tasks under the assumption of independent graphs in the training set [26], a condition not always met in link prediction. Considering these limitations, this work aims to develop an algorithm that ensures DP while maintaining a balance between utility and privacy, and is adaptable across various GNN modules.

III. Preliminaries and Problem Formulation

A. Notations

Let $G = (V, E)$ be an undirected graph where V is the set of nodes, E is the set of observed links, and $d(u, v)$ denotes the shortest path between a pair of nodes u and v. For a node u, its h-hop neighbors is denoted as $N_h(u) = \{v \mid d(u, v) = h, v \in V\}$, and its h-hop neighborhood is $\Gamma_h(u) = \cup_{i=0}^{h} N_i(u)$. Noted that nodes in $N_h(u)$ are exactly h hops away from u, and $N_0(u)$ denotes the node u itself.

Let $S = (V_S \subseteq V, E_S \subseteq E)$ be a subgraph of G, it can be extracted by a function $g : G \times \mathcal{E} \times \mathbb{N} \to S$. Here, $\mathcal{E}$ is an entity set, the elements of which can be either a vertex $u \in V$ or a pair of vertices $u, v \in V$. Any positive integer $h \in \mathbb{N}$ denotes the size of the extracted subgraph. Let $z \in \mathcal{E}$, S_z represents the enclosing subgraph that captures the local structure around z.

In addition to the concepts mentioned above, the main notations within this paper are listed in Table I.

TABLE I: Notations

Notation	Description
G	Undirected graph
V	Node set
E	Edge set
$\mathbf{A}$	Adjacency matrix
$\mathbf{X}$	Node feature matrix
$\mathcal{E}$	Entity set of a vertex or a pair of vertices.
S_z	Enclosing subgraph of z, where $z \in \mathcal{E}$.
$d(u, v)$	Shortest distance between u and v
$N_h(u)$	Node u's h-hop neighbors: $\{v \mid d(u, v) = h, v \in V\}$
$\Gamma_h(u)$	H-hop neighborhood of node u: $\cup_{i=0}^{h} N_i(u)$
$\Gamma_h(u, v)$	H-hop neighborhood of link (u, v): $\Gamma_h(u) \cup \Gamma_h(v)$
$LSN_h^g(s, q)$	Set of subgraph contextual neighbors of (s, q)

B. Graph Neural Networks

Let $G = (V, E)$ be an undirected graph with an adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$, where $n = |V|$. An r-layer GNN is a parametric function that can be represented by the following node-wise operations:

$$\begin{aligned} \mathbf{h}_u &= \mathrm{GNN}(\mathbf{A}, \mathbf{x}_u, \mathbf{\Theta}) \\ &= \psi\left(\mathbf{x}_u, \omega\left(\{\phi(\mathbf{x}_u, \mathbf{x}_v) \mid v \in \Gamma_r(u)\}\right)\right) \end{aligned} \tag{1}$$

where $\mathbf{x}_u \in \mathbb{R}^d$ and $\mathbf{h}_u \in \mathbb{R}^{d'}$ are d-dimensional input and d'-dimensional output features of node u, respectively. All functions ψ, ω, ϕ are learnable, and the parameters of the whole network are denoted as $\mathbf{\Theta}$. The above equation can be written in matrix representation as $\mathbf{H} = \mathrm{GNN}(\mathbf{A}, \mathbf{X}, \mathbf{\Theta})$ across all nodes, where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the node features, and $\mathbf{H} \in \mathbb{R}^{n \times d'}$ is the node representations.

C. Subgraph-based GNNs (SGNNs)

Given an undirected graph $G = (V, E)$ and a target $z \in \mathcal{E}$ (i.e., one or two nodes), SGNNs produce the embeddings $\mathbf{H}_z = \mathrm{GNN}(\mathbf{A}_z, \mathbf{X}_z, \mathbf{\Theta})$, where $\mathbf{A}_z$ presents the structure of the enclosing subgraph S_z, and $\mathbf{X}_z$ are the features of the nodes in S_z. Capturing the local structure, the embeddings can be effective for the downstream tasks.

In particular, for link prediction, the most common definition of the enclosing subgraph is the h-hop neighborhood subgraph. Its formal definition is as follows:

Definition 1 (H-Hop Neighborhood Subgraph)**.** *Given an undirected graph $G = (V, E)$ and a pair of distinct nodes $u, v \in V$, (u, v)'s h-hop neighborhood subgraph S_{uv} is the subgraph induced by the union of u's and v's h-hop neighborhood $\Gamma_h(u, v) = \Gamma_h(u) \cup \Gamma_h(v)$, i.e., $S_{uv} = g_n(G, (u, v), h)$, where g_n is the subgraph extraction function specific to the context of neighborhood.*

SGNNs can approximate global heuristics (i.e., graph structural features) from the h-hop neighborhood subgraph. The precision loss of this approximation diminishes exponentially with h [6].

D. Link-level Differential Privacy

This study aims to protect private links by applying differential privacy. As with the existing work [9], [27], we first define neighboring graphs and link differential privacy as follows.

Definition 2 (Neighboring Graphs)**.** *Two graphs G and G' are neighboring graphs if one can be obtained by adding or removing a link to the other.*

Definition 3 (Link Differential Privacy)**.** *A randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-link differential privacy, if and only if for any two neighboring graphs G, G', and any possible output $\mathcal{Z} \subseteq range(\mathcal{M})$, the following inequality holds:*

$$Pr(\mathcal{M}(G) \in \mathcal{Z}) \leq \varepsilon \cdot Pr(\mathcal{M}(G') \in \mathcal{Z}) + \delta.$$

Intuitively, a randomized algorithm $\mathcal{M}$ is said to be link-level differentially private if the addition or removal of a link in $\mathcal{M}$'s input does not affect $\mathcal{M}$'s output significantly. As shown below, the impact of the link difference between neighboring graphs influences the sensitivity analysis, which is critical to determine the noise scale in compliance with DP.

Definition 4 (Sensitivity under link differential privacy)**.** *The sensitivity $\Delta(f)$ of a function f defined on graph datasets is*

$$\Delta(f) = \max_{G,G'} \|f(G) - f(G')\|.$$

E. Problem Formulation

This work studies the problem of link prediction, which aims to predict the probability, denoted as $p(u, v)$, of a link between two nodes u and v. In this context, SGNNs have demonstrated effectiveness, and the link probability is computed based on the enclosing subgraph S_{uv}. Formally,

$$p(u, v) = R(\text{GNN}(\mathbf{A}_{uv}, \mathbf{X}_{uv}, \mathbf{\Theta})), \tag{2}$$

where R is a learnable readout function.

Our work focuses on the problem of designing an algorithm to learn GNNs while preserving link differential privacy. The difficulty in sensitivity analysis makes this problem non-trivial. Due to the data dependency issue in GNNs, the sensitivity requires a careful analysis of the maximum number of link embeddings that are affected by the removal of a link. This requires us to develop an analytical approach to derive a tight upper bound of this quantity to constrain the dependency level.

IV. Differentially Private Link Prediction

In this section, we propose a GNN-enabled framework for differentially private link prediction (DPLP). We first show an overview of the framework DPLP in Section IV-A, and then elaborate on the implementation details in Section IV-B.

A. DPLP Overview

DPLP framework is designed to train a link prediction model that is compatible with a wide range of GNN architectures, balancing differential privacy (DP) protection and model utility. DPLP works as shown in Figure 2, which includes three modules, namely subgraph extraction, sensitivity analysis, and DPGNN training. In what follows, we will present the design rationale of each module.

Subgraph Extraction. Following the success of SGNNs in capturing the local structure, DPLP incorporates SGNNs for link embedding and integrates a subgraph extraction module to support different subgraph extraction functions. Before subgraph extraction, DPLP includes a graph-specific projection step. This is because, the scale of DP noise increases exponentially with the maximum node degree, as will be elaborated in Lemma 3. To address the issue, this step projects the original graph to ensure a maximum node degree θ. In conjunction with the negative sampling method in subgraph extraction, this step efficiently reduces and bounds the noise scale.

Sensitivity Analysis. We incorporate a sensitivity analysis module to examine link dependency by introducing the concept of link-based subgraph contextual neighbor, ensuring a precise and optimized sensitivity measure for link differential privacy.

DPGNN Training. Gradient perturbation is a paradigm solution for training differentially private GNN, thanks to its architecture-agnostic nature which allows the algorithm to be applied across different GNNs and aggregation functions. However, conventional gradient perturbation methods, such as DP-SGD [25], are not directly applicable to GNNs. This arises from the inherent structure of GNNs where each per-sample (or per-link) gradient is influenced by private links from multiple users. This complexity poses challenges in determining an appropriate noise scale for achieving private gradients. In DPLP, this interdependency measured by the sensitivity analysis module allows us to accurately control the noise magnitude to derive the private gradient. Additionally, in Section VI, we will demonstrate that our perturbation approach also amplifies privacy.

Putting three modules together leads to our framework DPLP. As shown in Figure 2, the original graph $G = (V, E)$ is projected to graph G^θ with a maximum node degree θ (step (1)). For every edge in the projected graph G^θ, including those from negative sampling, their respective subgraphs are extracted. These subgraphs, paired with their labels, form the training dataset (step (2)). Given the subgraph extraction function g, and the subgraph size h, the maximum number of subgraph contextual neighbors of any link in G is computed, represented as $\max_{(s,q)\in E} |LSN_h^g(s, q)|$ (step (3)). Then, the sensitivity is derived by this bound (step (4)). The progression of training the differentially private GNN is presented in steps (5) - (9). Initially, a batch of subgraphs is uniformly sampled from the training set and fed into the GNN (step (5)). The gradient perturbation mechanism involves two steps, namely clipping per-sample gradients (step (6)) and injecting noise into the batched gradient to yield a private gradient (step (8)), where the noise is sampled according to the pre-defined sensitivity (step (7)). This sequence, spanning from step (5) to step (9), is iterated until the stop condition is achieved. Finally, DPLP outputs a trained GNN model preserving link differential privacy.

B. DPLP Implementation

In this section, we shows the implementation details of DPLP. In particular, we first present subgraph extraction and DPGNN training, followed by sensitivity analysis which intro-

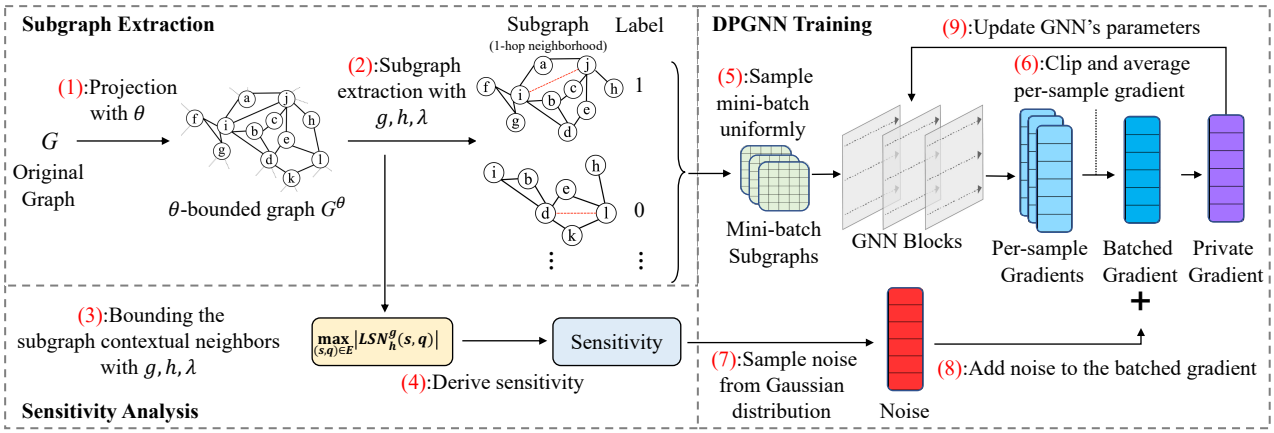


Fig. 2: The overview of the DPLP framework.

duces the notion of link-based subgraph contextual neighbor and gives the theoretical results.

1) Subgraphs Extraction: Given a maximum node degree of θ, the graph projection step projects the original graph into G^θ so that the degree of each node is bounded by θ. As outlined in Algorithm 1 (Lines 9-15), for each node v, we randomly sample at most θ immediate neighbors from E^θ of G^θ, and update the edges in E^θ by dropping the edges between v and the excessive neighbors. The parameter θ decides the number of available samples for the training module and the scale of influence when a link is removed, which in turn affects the sensitivity of DP noise. In Section 5.2, we will delve into the selection of this parameter.

Then the subgraph extraction step extracts a set of subgraphs along with their node labels, constituting training dataset for link prediction model. The purpose is to acquire the local structure of the target link to be classified. Specifically, for each link in projected graph G^θ, we will extract a corresponding subgraph, which serves as a positive sample (i.e., an observed link in the graph) in training dataset. To prevent the model bias, it is necessary to include negative samples as well. To achieve this, we will also sample some non-existent links and extract their subgraphs.

Algorithm 1 demonstrates the neighborhood subgraph extraction process. Note that for each node v, there are $|N_1(v)|$ positive links. Thus, given the negative sampling rate λ, $\lambda|N_1(v)|$ negative links will be sampled from $V \backslash N_1(v)$ (Line 3). Then for each positive or negative link, a subgraph will be extracted by the subgraph extraction function g_n (Line 6), along with a binary label indicating whether it is a positive or negative sample (Line 7).

2) Training Differentially private GNNs: Learning a GNN is equivalent to finding optimal parameters $\mathbf{\Theta}^*$ that minimizes the loss function $\mathcal{L}(\mathcal{S}, \mathbf{\Theta})$. Formally

$$\begin{aligned}\mathbf{\Theta}^* &= \arg\min_{\mathbf{\Theta}} \mathcal{L}(\mathcal{S}, \mathbf{\Theta}) \\ &= \arg\min_{\mathbf{\Theta}} \sum_{S_{uv}, y_{uv} \in \mathcal{S}} \ell\left(R\left(\mathrm{GNN}\left(\mathbf{A}_{uv}, \mathbf{X}_{uv}, \mathbf{\Theta}\right)\right), y_{uv}\right),\end{aligned} \tag{3}$$

Algorithm 1: EXTRACT-SUBGRAPHS (Extract Neighborhood Subgraphs)

Data: θ-bounded graph $G^\theta = (V, E^\theta)$; Negative sampling rate λ; Neighborhood radius h; Neighborhood subgraph extraction function g_n.
Result: Set of subgraphs S_{uv} and lables y_{uv} for target links.

1 $G^\theta \leftarrow \mathrm{PROJECTION}(G, \theta)$
2 **for** $v \in V$ **do**
3 Uniformly sample negative edges of v: $\bar{N}_1(v) \leftarrow \mathrm{sample}(V \backslash N_1(v), \lambda|N_1(v)|)$
4 Add $\bar{N}_1(v)$ to the set of negative edges NE
5 **for** $i \in N_1(v) \cup \bar{N}_1(v)$ **do**
6 Extract subgraph: $S_{vi} \leftarrow g_n(G^\theta, (v, i), h)$
7 $y_{vi} = \begin{cases} 1 & \text{if } i \in N_1(v) \\ 0 & \text{otherwise} \end{cases}$
8 **return** $\{S_{uv}, y_{uv} \mid (u, v) \in E^\theta \cup NE\}$.
9 **Function** $\mathrm{PROJECTION}(G, \theta)$**:**
10 Initialize $G^\theta = (V^\theta, E^\theta) \leftarrow G$
11 **for** $v \in V$ **do**
12 $N_1^\theta(v) \leftarrow$ Randomly sample at most θ immediate neighbors of v by E^θ.
13 $e = \{(u, v) \mid u \in \{N_1(v) \backslash N_1^\theta(v)\}\}$
14 Update E^θ by dropping edges in e from E^θ.
15 **return** $G^\theta = (V^\theta, E^\theta)$

where $\mathcal{S}$ is the training subgraphs returned by Algorithm 1. The learning algorithm towards the above loss is summarized in Algorithm 2. First, a training set is established on the projected graph (Lines 1-2). Next, we iteratively update the GNN model after model initialization. In iteration t, a batch $\mathcal{B}_t$ of m subgraphs is uniformly sampled from $\mathcal{S}$. It guarantees each subgraph appears only one time within $\mathcal{B}_t$. Subsequently, per-sample gradients are computed and clipped (Lines 7-10) with a given l^2 norm threshold C. Gaussian noise is added to the batched gradient that is aggregated from clipped per-sample gradients (Lines 11-12). The private gradients are

employed for model updating. This process continues until the maximum number of iterations T is reached.

Algorithm 2: TRAIN-DPGNN (Neighborhood Subgraph-based DPGNN)

Data: Graph $G = (V, E)$; node features matrix $\mathbf{X}$; Maximum node degree θ; Negative sampling rate λ; Neighborhood radius h; Neighborhood subgraph extraction function g_n; Maximum number of iterations T.

Result: The trained model $\mathbf{\Theta}_T$

$G^\theta \leftarrow \text{PROJECTION}(G, \theta)$

Construct the set of training subgraphs and labels: $\mathcal{S} \leftarrow$ EXTRACT-SUBGRAPHS$(G^\theta, r, \lambda, h, g_n)$

Initialize $\mathbf{\Theta}_0$ randomly

for $t = 0$ *to* T **do**

  Sample set $\mathcal{B}_t \subseteq \mathcal{S}$ of size m uniformly

  for *each* (S_{uv}, y_{uv}) *in* $\mathcal{B}_t$ **do**

    Compute gradient:

    $\mathbf{g}_t(S_{uv}, y_{uv}) \leftarrow \nabla_{\mathbf{\Theta}} \ell\left(R\left(\text{GNN}\left(\mathbf{A}_{uv}, \mathbf{X}_{uv}, \mathbf{\Theta}_t\right)\right), y_{uv}\right)$

    Clip gradient:

    $\hat{\mathbf{g}}_t(S_{uv}, y_{uv}) \leftarrow \mathbf{g}_t(S_{uv}, y_{uv}) / \max\left(1, \frac{\|\mathbf{g}_t(S_{uv}, y_{uv})\|_2}{C}\right)$

  $\overline{\mathbf{g}}_t \leftarrow \sum_{S_{uv}, y_{uv} \in \mathcal{B}_t} \hat{\mathbf{g}}_t(S_{uv}, y_{uv})$

  Add noise: $\tilde{\mathbf{g}}_t \leftarrow \overline{\mathbf{g}}_t + \mathcal{N}\left(0, \sigma^2 \mathbb{I}\right)$

  Update the parameters: $\mathbf{\Theta}_{t+1} \leftarrow \mathbf{\Theta}_t - \frac{\eta}{m} \overline{\mathbf{g}}_t$

3) Sensitivity Analysis: For sensitivity analysis in subgraph-based GNNs, understanding link dependency is crucial. While node-level tasks leverage neighboring relationships to measure node dependency, no such standard exists for links. Consequently, we introduce the notion of link-based subgraph contextual neighbor to describe the interdependency between links.

Definition 5 (Link-based Subgraph Contextual Neighbor)**.** *Given an undirected graph $G = (V, E)$, a pair of nodes $u, v \in V$, the subgraph size h and a subgraph extraction function g. If the subgraph $g(G, (u, v), h)$ contains a link (s, q), then (u, v) is a subgraph contextual neighbor of (s, q), which is denoted as $(u, v) \in LSN_h^g(s, q)$.*

Given a graph $G = (V, E)$ and a neighborhood subgraph extraction function g_n, let $LSN_h^{g_n}(u, g)$ denote the number of subgraph contextual neighbors of any link $(u, g) \in E$. The upper bound of this number, denoted as $\max_{(s,q)\in E} |LSN_h^{g_n}(u, g)|$, is the building block of sensitivity analysis module. The following Lemma 1 shows the derivation of $LSN_h^{g_n}(s, q)$, and then Lemma 2 provides its upper bound. Based on the derived bound, the sensitivity of adding or removing a link is given by Lemma 3.

Lemma 1. *Consider the following link sets after the graph projection and negative sampling:*

- $e_1 = \{(u, v) \mid u \in \Gamma_h(s) \cap \Gamma_h(q), v \in N_1(u) \cup \bar{N}_1(u)\}$;
- $e_2 = \{(u, v) \mid u \in \Gamma_h(s) \backslash \Gamma_h(q), v \in \Gamma_h(q) \backslash \Gamma_h(s)\}$.

Then, $LSN_h^{g_n}(s, q) = e_1 \cup e_2$.

Lemma 2. *Let G be any graph with a set of training edges E, λ be the negative sampling rate for each node, $\theta \geq 2$ be the maximum node degree, and $h \geq 1$ be the radius of the neighborhood. Then, $\max_{(s,q)\in E} |LSN_h^{g_n}(s, q)|$ is upper bounded by $M_{g_n}(h, \theta, \lambda)$, where*

$$M_{g_n}(h, \theta, \lambda) = (1 + \lambda)\left(\theta^{h+1} + \sum\nolimits_{i=0}^{h-1} 2\theta^{i+1}\right) \quad (4)$$

Proof. Let $I_{ij} = N_i(s) \cap N_j(q)$, where $0 \leq i, j \leq h$, $e_1^i(s) = \{(u, v) \mid u \in \bigcup_{j=i}^h I_{ij}, v \in N_1(u) \cup \bar{N}_1(u)\}$, $e_1^h(s, q) = \{u \in I_{hh}, v \in N_1(u) \cup \bar{N}_1(u)\}$. Then, $e_1 = e_1^h(s, q) \bigcup_{i=0}^{h-1} \left(e_1^i(s) \cup e_1^i(q)\right)$. Let $P = (\Gamma_h(s) \backslash \Gamma_h(q)) \cap (\Gamma_h(q) \backslash \Gamma_h(s))$. Then, the following inequality holds:

$$\begin{aligned} &|e_2| + |e_1^h(s, q)| \\ \leq &\max_P \left\{|P|(1 + \lambda)\theta + (\theta^h - |P|) \min\left\{(1 + \lambda)\theta, \theta^h - |P|\right\}\right\} \\ \leq &\ \theta^h(1 + \lambda)\theta, \end{aligned}$$

$$\begin{aligned} |e_1^i(s)| &\leq \left|\bigcup\nolimits_{j=i}^h I_{ij}\right| \cdot \min\{|N_1(u) \cup \bar{N}_1(u)|, \theta(1 + \lambda)\} \\ &\leq |N_i(s) \backslash \Gamma_i(q)| \cdot \theta(1 + \lambda) \leq \theta^{i+1}(1 + \lambda). \end{aligned}$$

Therefore, we have the following inequality:

$$\begin{aligned} \left|\bigcup\nolimits_{i=0}^{h-1} \left(e_1^i(s) \cup e_1^i(q)\right)\right| &\leq \sum\nolimits_{i=0}^{h-1} \left(|e_1^i(s)| + |e_1^i(q)|\right) \\ &\leq \sum\nolimits_{i=0}^{h-1} 2\theta^{i+1}(1 + \lambda) \end{aligned}$$

Combine the former results, the maximum number of subgraph contextual neighbors of (s, q) is bounded by

$$\begin{aligned} |LSN_h^{g_n}(s, q)| &\leq |e_1| + |e_2| \\ &\leq |e_1^h(s, q)| + \left|\bigcup\nolimits_{i=0}^{h-1} \left(e_1^i(s) \cup e_1^i(q)\right)\right| + |e_2| \\ &\leq (1 + \lambda)\left(\theta^{h+1} + \sum\nolimits_{i=0}^{h-1} 2\theta^{i+1}\right) \end{aligned}$$

Let $M_{g_n}(h, \theta, \lambda)$ be the upper bound, then the proof is completed. □

Lemma 3 (Link-level Sensitivity of Neighborhood Subgraph-based GNN.)**.** *Let G^θ be a graph with bounded degree θ, λ be the negative sampling rate, h be the radius of the neighborhood, $\mathcal{B}_t$ be the batch of subgraphs at any t step in Algorithm 2. Considering the batched gradient $\overline{\mathbf{g}}_t$ obtained from aggregating the gradient clipped by parameter C. Then the following inequality holds:*

$$\Delta(\overline{\mathbf{g}}_t) < 2C \cdot (1 + \lambda)\left(\theta^{h+1} + \sum\nolimits_{i=0}^{h-1} 2\theta^{i+1}\right) \quad (5)$$

V. DPLP WITH PATH SUBGRAPH EXTRACTION

The sensitivity analysis in Section IV reveals that, the number of subgraph contextual neighbors relies on the subgraph extraction process. While the h-hop neighborhood subgraph

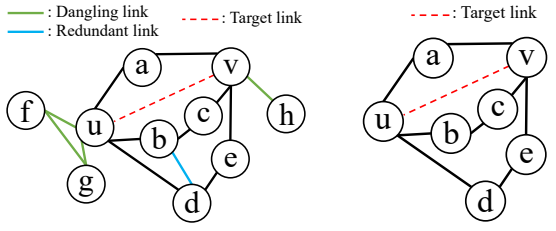
is effective at learning diverse heuristic methods for link prediction [6], the number of subgraph contextual neighbors increases exponentially with h (see Equation 4), leading to a significant noise sensitivity. Therefore, there's a need for a method that reduces the number of subgraph contextual neighbors while still retaining the ability to effectively learn informative heuristics. In this section, we introduce such a balanced subgraph extraction approach.

A. Path Subgraph

To reduce the noise sensitivity, in this section we propose an optimized path subgraph extraction approach, which exclusively extracts the links on paths between a pair of nodes. Formally, a k-hop path subgraph is defined as follows:

Definition 6 (K-Hop Path Subgraph)**.** *Given an undirected graph $G = (V, E)$ and a pair of distinct nodes $u, v \in V$, let P^j_{uv} be the set of paths from u to v with length j. For any link (u, v), the k-hop path subgraph S_{uv} consists of all the links $E_{uv} = \bigcup_{j=2}^{k} \bigcup_{w \in P^j_{uv}} \bigcup_{i=0}^{j-1} \{(w_i, w_{i+1})\}$.*

Figure 3 illustrates a comparison of neighborhood and path subgraphs. Figure 3(a) showcases the neighborhood subgraph for the link (u, v) when $h = 1$, which corresponds to a path subgraph with $k \leq 3$ in Figure 3(b). In the path subgraph, only the paths between (u, v) in the neighborhood are included, while redundant links (marked by blue links) and dangling links (marked by green links) are removed. Redundant links, which are induced by neighborhood nodes, fall out of any path from u to v with a length up to k. Dangling links connect dangling nodes, which are nodes that can only reach one of the vertices (u, v) within a 1-hop distance. Obviously, both redundant and dangling links contribute less to the relationship between u and v, but increasing the noise sensitivity. Removing them streamlines hierarchical analysis of subgraph contextual neighboring links based on path length. Consequently, these links, as illustrated in Figure 3, are isolated from the path between u and v.



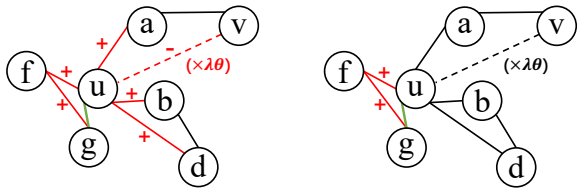
(a) 1-hop neighborhood subgraph (b) 3-hop path subgraph

Fig. 3: Examples of neighborhood and path subgraphs

By excluding dangling links in the subgraph of a link, the path subgraph effectively reduces the number of subgraph contextual neighbors. The examples provided in Figure 4 clearly demonstrate our rationale for this approach.

Let's consider the impact of adding a link (u, g) to the graphs in Figure 4, where the maximum node degree $\theta = 5$. The link-based subgraph contextual neighbors of the link (u, g) in each subfigure are highlighted in red (excluding (u, g) itself). The red solid lines represent observed links, a.k.a. positive links, while the red dashed lines indicate negative links after negative sampling with a rate of λ. Figure 4(a) shows the case of extracting neighborhood subgraphs, while Figure 4(b) demonstrates the case of extracting path subgraphs.

Given the neighborhood radius $h = 1$, the neighborhood and path subgraph extraction functions are denoted as g_n and g_p, respectively. As shown in Figure 4(a), $\{(a, u), (b, u), (d, u), (f, u), (f, g)\} \subset LSN_1^{g_n}(u, g)$. And $\lambda\theta$ negative links incident to u, e.g., (u, v), are also encompassed within $LSN_1^{g_n}(u, g)$. This is because all these links fall into the set e_1 with $h = 1$ defined in Lemma 1, where $u \in N_0(u) \cap N_1(g)$ and $g \in N_0(g) \cap N_1(u)$. By comparison, as shown in Figure 4(b), when extracting the path subgraph from the same graph, $LSN_3^{g_n}(u, g) = \{(f, u), (f, g)\}$. This is because (g, u) is only on the paths from node f to g and u.



(a) Extracting Neighborhood Subgraphs (b) Extracting Path Subgraphs

Fig. 4: Link-based subgraph contextual neighbors of link (u, g) in neighborhood and path subgraphs, when $h = 1$

The following theorem establishes the theoretical analysis on data dependency of the proposed path-based subgraph extraction approach.

Theorem 1. *Given an undirected graph $G = (V, E)$, let λ be the negative sampling rate for each node and θ be the maximum node degree. For any $k, \theta \geq 2$, $\max_{(s,q)\in E} |LSN_h^{g_p}(s, q)|$ is bounded by $M_{g_p}(k, \theta, \lambda)$, where*

$$M_{g_p}(k, \theta, \lambda) = \sum_{i=0}^{h-1} 2\lambda\theta^{i+1} + \theta^h \cdot \min\{(1+\lambda)\theta, \theta^h\} + 2\sum_{i=1}^{h} \min\{\theta^{h+1}, \theta^i\}, \quad (6)$$

and $h = \lceil \frac{k-1}{2} \rceil$. Particularly, when $k = 2$, $M(k, \theta, \lambda) = 2\theta$.

Proof. According to the Lemma 1, for any link (u, v) in the contextual neighbors $LSN_h^{g_n}(s, q)$, at least one vertex is in $\Gamma_h(s, q)$. By the definition of the path subgraph, when $k \leq 2h + 1$, the k-hop path subgraph is a subgraph of the h-hop neighborhood subgraph, and $LSN_k^{g_p}(s, q)$ is a subset of $LSN_h^{g_n}(s, q)$. Therefore, for any link (u, v) in $LSN_k^{g_p}(s, q)$, at least one node is in $\Gamma_h(s, q)$. Thus, when $LSN_h^{g_p}(s, q)$ is maximized, the number of nodes in $\Gamma_h(s, q)$ is also maximized. And we have $|\Gamma_{h-1}(s, q)| \leq \sum_{i=0}^{h-1} 2\theta^i$ and $\Gamma_{h-1}(s) \cap \Gamma_{h-1}(q) = \emptyset$. Under such conditions, there are no cycles in the $(h-1)$-hop neighborhood subgraph of (s, q). Consequently, due to the exclusion of all dangling links, the maximum number of connected nodes (i.e., $(\lambda + 1)\theta$) cannot be achieved for dangling nodes (i.e., $\Gamma_{h-1}(s, q)$). Thus, the

maximum number of links incident to each node in $\Gamma_{h-1}(s,q)$ is $\lambda\theta$. This leads to the following inequality:

$$\max_{(s,q)\in E} |LSN^{g_p}_{k-1}(s,q)| \leq \sum\nolimits_{i=0}^{h-1} 2\lambda\theta^{i+1} \tag{7}$$

Now, we consider the contextual neighbors related to the h-hop neighbors. Let

$$A = \{(x,y) \mid x \in \hat{N}_h(s)\backslash\hat{N}_h(q), y \in \hat{N}_h(q)\backslash\hat{N}_h(s)\},$$

where $\hat{N}_h(s) = N_h(s)\backslash\Gamma_{h-1}(q)$, and $\hat{N}_h(q) = N_h(q)\backslash\Gamma_{h-1}(s)$. Noted that there are no links containing any node in $\hat{N}_h(s)\backslash\hat{N}_h(q)$, since the connecting node can visit neither s nor q within $\lfloor(k+1)/2\rfloor$ hops. Since there are no neighbors whose hop is higher than $\lfloor(k+1)/2\rfloor$, the nodes in A can form positive or negative links to reach the maximum number of dependent links by

$$|A| \leq \theta^h \cdot \min\{(1+\lambda)\theta, \theta^h\} \tag{8}$$

Combining the two upper bounds in Equations 7 and 8, we can get Equation 6. □

Remark. *Based on Theorem 1, we can derive the upper bound on the reduction in the number of dependent links achievable by the path subgraph. Let $h \geq 1$ be the radius of the neighborhood, $k = 2h+1$ be the maximum length of a path in the h-hop neighborhood subgraph, $\theta > 2$ be the maximum node degree, and $\lambda \geq 1$ be the negative sampling rate. The number of subgraph contextual neighbors in the path subgraph can be reduced by is shown below:*

$$M_{g_n}(h,\theta,\lambda) - M_{g_p}(k,\theta,\lambda) = \begin{cases} \lambda\theta^2 + 2\theta & h = 1 \\ \frac{2(\theta^{h+1}-\theta)}{\theta-1} & h > 1 \end{cases} \tag{9}$$

B. Parameter selection

In this section, we design an indicator to better determine the parameters k and θ by considering the loss introduced by projection and gradient perturbation. Let $c \in \mathbb{R}^n$ be the degree distribution of a graph G, where n is the number of unique degrees, and each c_i denotes the number of nodes with degree i. Since c is computed on G, it should be perturbed by DP as well. Let c, c' denote the degree distribution computed on the neighboring graphs $G \simeq G'$ that differ in one link, the sensitivity can be bounded by

$$\|c - c'\|_2 \leq \sqrt{2}. \tag{10}$$

The private version of c is obtained by adding Gaussian noise as follows:

$$\tilde{c} = c + \mathcal{N}(0, \sigma^2\mathbb{I}). \tag{11}$$

The loss caused by projection can be approximated as

$$\ell_{proj}(\theta, \tilde{c}) = \sum\nolimits_{i=d^*}^{n} \tilde{c}_i \tag{12}$$

where d^* is the smallest degree that is greater than θ, i.e., $d^* = \min\{i \mid i > \theta, i \in \{1,2,\cdots,n\}\}$. Given any k and θ, we design an indicator as $\frac{1}{\ell(k,\theta)}$, where $\ell(k,\theta)$ is to capture the projection and perturbation loss, which is defined as

$$\ell(k,\theta) = \alpha \cdot \frac{|E|}{(1-\gamma^k)(|E| - \ell_{proj})} + \beta \cdot \sigma_c(\theta, k, \varepsilon) \tag{13}$$

Note that in Equation 13, $|E|$ is the number of observed links in the original graph, and σ_c computes the noise multiplier with a given privacy budget ε for training DPGNN. The first term computes the ratio of the total links to the remaining links after projection with a decaying factor $\gamma \in (0,1)$ weighting the length of the path. The second terms approximates the magnitude of perturbation during DPGNN training. Two parameters α and β weight these two parts. Using this indicator, we can more adeptly choose optimal values for k and θ to minimize the overall loss introduced by the projection and perturbation process. The efficacy of this indicator will be further evaluated in Section VII.

VI. Theoretic Analyses

In this section, we establish theoretical analysis of the DPLP framework in terms of privacy and utility guarantee.

A. Privacy Analysis

The following theorem shows that DPLP achieves Link DP by offering $(\alpha, \varepsilon^\rho(\alpha))$-Rényi differential privacy. Here, according to [28], we use the functional view of Rényi differential privacy in which ε is a function of α, where $1 < \alpha < \infty$, and this function is determined by the private algorithm.

Theorem 2. *Let N be the number of training subgraphs, m be the batch size, g be the subgraph extraction function, h be the size of subgraphs, λ be the negative sampling rate, and θ be the maximum node degree, $M_g(h,\theta,\lambda)$ be the upper bound of subgraph contextual neighbors. Then, every iteration t of Algorithm 2 satisfies $(\alpha, \varepsilon^\rho(\alpha))$-Rényi DP, where*

$$\begin{aligned}\varepsilon^\rho(\alpha) \leq \frac{1}{\alpha-1}\log\Bigg(1 + \rho^2\binom{\alpha}{2}\min\Big\{4\left(e^{\varepsilon(2)}-1\right), \\ e^{\varepsilon(2)}\min\left\{2, \left(e^{\varepsilon(\infty)}-1\right)^2\right\}\Big\} \\ + \sum\nolimits_{j=3}^{\alpha}\rho^j\binom{\alpha}{j}e^{(j-1)\varepsilon(j)}\min\left\{2, \left(e^{\varepsilon(\infty)}-1\right)^j\right\}\Bigg),\end{aligned} \tag{14}$$

and

$$\rho = 1 - \binom{N - M_g(h,\theta,\lambda)}{m} \Big/ \binom{N}{m}. \tag{15}$$

By the standard composition theorem of Rényi Differential Privacy [29], over T iterations, Algorithm 2 is $(\alpha, \varepsilon^\rho(\alpha)T)$-Rényi DP.

Proof. Without considering privacy amplification, every iteration t of Algorithm 2 is $(\alpha, \varepsilon(\alpha))$-Rényi DP where $\varepsilon(\alpha) = \frac{\alpha\cdot(\Delta(\overline{\mathbf{g}}_t))^2}{2\sigma^2}$. This can be directly derived from Corollary 3 in [29]. Let $E_{\mathcal{B}_t}$ be all links in the batched subgraphs, the

probability that any specific link $(s,q) \in E$ is contained in $E_{\mathcal{B}_t}$ can be bounded by

$$\max_{(s,q)\in E} Pr\left((s,q) \in E_{\mathcal{B}_t}\right) \leq 1 - \left(\begin{array}{c} N - M_g(h,\theta,\lambda) \\ m \end{array} \right) \Big/ \left(\begin{array}{c} N \\ m \end{array} \right). \quad (16)$$

By setting ρ to this bound, the privacy guarantee of Algorithm 2 can be directly analyzed by Theorem 9 of [28], yielding a Rényi DP of $(\alpha, \varepsilon^{\rho}(\alpha))$ with $\varepsilon^{\rho}(\alpha)$ defined in Equation 14. □

Notably, the key difference of Theorem 2 from the unamplified Rényi differential privacy [29] is the introduction of ρ (as shown in Equation 15) to calculate the final privacy parameter. A less number of subgraph contextual neighbors leads to a smaller ρ, indicating a lower probability of including dependent subgraphs within each training batch. This reduction in ρ can enhance the effect of privacy amplification, which is the distinct advantage of the path subgraph method. Finally, we apply the conversion rule in [29] to convert the Rényi differential privacy back to link DP.

B. Utility Analysis

Theorem 1 proves the advantage of the path subgraph compared to the neighborhood subgraph in terms of noise sensitivity. This improvement is primarily attributed to the fact that, when extracting the neighborhood subgraphs, many dangling links are included in sensitivity analysis as worst-case scenarios, leading to an increasing number of dependent links. By limiting the count of dangling edges, the sensitivity can be notably reduced.

Although the path subgraph effectively reduces the number of subgraph contextual neighbors, it unavoidably results in information loss when edges are dropped. One crucial piece of information missing in the path subgraph is the real node degrees. Degree information is widely utilized in local heuristic methods, especially those that rely on popularity measures (e.g., Preferential Attachment [30]). Nevertheless, there are still several k-hop path heuristic methods that can be learned from path subgraphs. The k-hop path heuristic methods solely rely on predicting based on paths between nodes (u,v) with a maximum length of k. Examples of such methods include Common Neighbors, Local Path Index and Katz Index, etc. Let S_{uv} denote the subgraph extracted by function g_p, we can establish the following Theorem.

Theorem 3. *Given a graph $G = (V,E)$ with maximum node degree D, let G^{θ} be the θ-bounded graph after projection with a threshold $\theta \leq D$. Let N be the number of training subgraphs, and M_{g_p} be the upper bound of the maximum number of path subgraph contextual neighbors in G. When $\theta \to D, (M_{g_p}-1)/N \to 0$, k-hop path heuristic score for (u,v) can be accurately approximated from the k-hop path subgraph S_{uv} around (u,v) and the approximation error decreases at least exponentially with k.*

Proof. According to [30], any k-hop path heuristics can be reduced to Katz Index by adjusting the path length k and a damping factor. Thus, we can complete the proof by proving the heuristic method of Katz Index. And the proof extends to all k-hop path heuristics. Let P^{k}_{xy}, $P^{k,\theta}_{xy}$ be the set of path with maximum length k on graph G and G^{θ}, respectively.

$$\mathrm{Katz}_{u,v} = \sum\nolimits_{l=1}^{\infty} \beta^l |P^l_{uv}|,\ \widetilde{\mathrm{Katz}}_{u,v} = \sum\nolimits_{l=1}^{g(k)} \beta^l |P^{l,\theta}_{uv}| \quad (17)$$

where $g(k) = ak + b$, with $a, b \in \mathbb{N}$ and $a > 0$. Then the approximation error between $\widetilde{\mathrm{Katz}}_{u,v}$ and $\mathrm{Katz}_{u,v}$ can be bounded as follows:

$$\begin{aligned} &\lim_{\theta\to D} |\,\mathrm{Katz}_{u,v} - \widetilde{\mathrm{Katz}}_{u,v}| \\ &= \lim_{\theta\to D} \sum\nolimits_{l=1}^{g(k)} \beta^l (|P^l_{uv}| - |P^{l,\theta}_{uv}|) + \sum\nolimits_{l=g(k)+1}^{\infty} \beta^l |P^l_{uv}| \\ &\leq \sum\nolimits_{l=ak+b+1}^{\infty} \beta^l \theta^l = (\beta\theta)^{ah+b+1}(1-\beta\theta)^{-1}. \end{aligned} \quad (18)$$

When, $(M_{g_p}-1)/N \to 0$, $\rho = 0$, and the noise multiplier converges to zero, leading to a GNN model without perturbation loss. In such condition, the GNN preserves the most expressive power and the Katz Index with any maximum path length k can be approximated from S_{uv} with error decrease at least exponentially with k. Since any k-hop path heuristics can be reduced to the Katz Index, the proof is complete. □

Theorem 3 demonstrates that by analyzing local subgraphs surrounding links, the expressiveness of the private path subgraph-based method converges to its non-private version when the training set of subgraphs is sufficiently large with small number of subgraph contextual neighbors. Thus, the path subgraph-based method has the potential to precisely learn the first and second-order path-based heuristics. Moreover, a broad spectrum of higher-order path-based heuristics can be estimated from path subgraph with fewer hops with minimal error.

VII. Experimental Evaluations

In this section, we conduct extensive experiments to empirically evaluate DPLP's performance in terms of privacy, accuracy, and the effectiveness of parameter selection.

A. Experimental Setting

Dataset. We evaluate the proposed methods on several publicly available link prediction datasets, by considering varying degree distributions, and the numbers of nodes and edges.

- USAir [1] is a network of US Airlines with 332 nodes and 2,126 edges. The average node degree is 12.81.
- PB [2] is a network of US political blogs with 1,222 nodes and 16,714 edges. The average node degree is 27.36.

[1] http://vlado.fmf.uni-lj.si/pub/networks/data/mix/USAir97.net
[2] https://sites.cc.gatech.edu/dimacs10/archive/clustering.shtml

- Yeast [3] is a protein-protein interaction network in yeast with 2,375 nodes and 11,693 edges. The average node degree is 9.85.
- C.ele [4] is a neural network of C. elegans with 297 nodes and 2,148 edges. The average node degree is 14.46.

Experiment Design. We implement DPLP with neighborhood subgraph extraction as our baseline, referred to as DPLP-NS. Next, we optimize the subgraph extraction module of DPLP by implementing path subgraph extraction, denoted as DPLP-PS. We then compare our methods with two other techniques: SEAL [6], which represents a non-private GNN method for link prediction, and LapGraph [9], which is a state-of-the-art differentially private technique designed for graph data.

Implementation and Parameters. For DPLP methods (DPLP-NS and DPLP-PS), we use GCN as the default architecture of SGNN, and we set the number of SGNN layers to k for DPLP-PS and $h = \lfloor (k+1)/2 \rfloor$ for DPLP-NS. For the LapGraph method, we use the SGNN based on the neighborhood subgraph as its backbone for link prediction. Similar to DPLP-NS, the number of layers is set to $h = \lfloor (k+1)/2 \rfloor$. For all methods, we set the number of hidden units to 32, and use the Relu activation function at every layer. Dropout layers are used for all methods. We train all methods over 50 epochs with a batch size of 1024 on PB, and 128 on the rest 3 datasets, i.e., USAir, Yeast and Celegans. For DPLP-NS, DPLP-PS, we use the same parameter setting for training GNN modules. We train all the methods with a learning rate of 0.01 and repeat each combination of possible hyperparameter values 10 times.

Performance Measurement. Area Under the Curve (AUC) is a commonly used performance measurement of a binary classification model [31]. AUC ranges from 0 to 1, with a higher value indicating better model performance. In the experiment, we pick the best-performing model based on validation AUC and report the average test AUC with standard error.

Privacy setting. We numerically calibrate the noise scale (i.e., the noise standard deviation σ divided by the sensitivity) of the DPLP-NS and DPLP-PS methods and the Gaussian mechanism to achieve the desired (ε, δ)-Link DP with 0.1ε for parameter selection. We report results for several values of ε, while δ is set to be smaller than the inverse number of private entities (i.e., links for link-level privacy).

All the models are implemented in PyTorch using PyTorch-Geometric (PyG). Experiments are conducted on two devices with NVIDIA GeForce RTX 4090 GPUs, Intel Xeon 6238 CPUs, and 32 GB RAM.

B. Experimental Results and Analysis

1) Overall Results: We first compare the AUC of our proposed methods against the non-private and the link-level private baseline (i.e., LapGraph). We fix the privacy budget to $\varepsilon = 4$ for the private methods. The results are presented in Table II. We observe that the proposed path subgraph-based method (DPLP-PS) is competitive with SEAL, with a decrease within 6.5% in AUC on the four datasets. DPLP-PS significantly outperforms the private competitors LapGraph and DPLP-NS. Particularly, DPLP-PS's AUC is roughly 37, 16, 25 and 5 points higher than LapGraph over Yeast, USAir, C.ele and PB, respectively.

TABLE II: **Test AUC(%) of different methods with $\varepsilon = 4$. The best-performing private method is highlighted.**

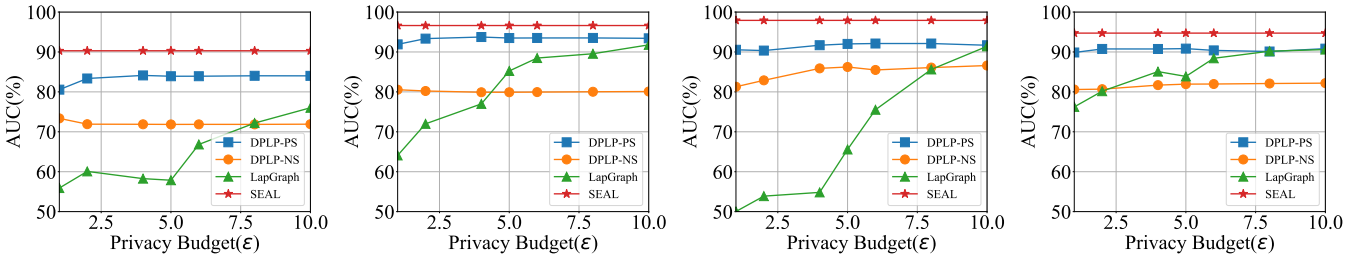
Model	Yeast	USAir	C.ele	PB
SEAL	97.91 ± 0.52	96.62 ± 0.72	90.30 ± 1.35	94.72 ± 0.46
LapGraph	54.82 ± 8.90	76.99 ± 11.18	58.26 ± 15.25	85.08 ± 5.24
DPLP-NS	85.89 ± 7.26	79.90 ± 14.09	71.88 ± 5.94	81.69 ± 11.66
DPLP-PS	$\mathbf{92.02 \pm 1.18}$	$\mathbf{93.74 \pm 0.04}$	$\mathbf{84.12 \pm 0.29}$	$\mathbf{90.75 \pm 0.24}$

The DPLP-NS model performs better than LapGraph on the Yeast, USAir, and C.ele datasets, demonstrating the argument that our gradient-perturbation paradigms are superior to input perturbation methods under strict privacy requirement. However, on the PB dataset, LapGraph achieves a higher AUC by approximately 4 points compared to DPLP-NS. This discrepancy can be attributed to the high average degree in PB, necessitating a higher θ to retain sufficient training samples. Nevertheless, an excessively large θ in DPLP-NS results in high sensitivity, thereby impacting the model's performance-a phenomenon not significantly observed in DPLP-PS. This results substantiates our theoretical findings that DPLP-PS enhances the model performance of privacy models on dense datasets by learning with lower sensitivity.

To study the impact of different privacy budgets on the performance of link prediction, we present the AUC for each method by varying ε from 1 to 10. The results are illustrated in Figure 5. It's obvious that DPLP-PS consistently outperforms LapGraph and DPLP-NS, particularly in more strict privacy settings (i.e., when given a small privacy budget). Specifically, DPLP-NS converges to its optimal value faster, yielding a superior AUC than LapGraph when the privacy budget is stringent (i.e., $\varepsilon \leq 2.5$). However, due to the larger noise added by DPLP-NS, it has a larger standard deviation, making its performance less stable. As the privacy budget increases, it is eventually surpassed by LapGraph.

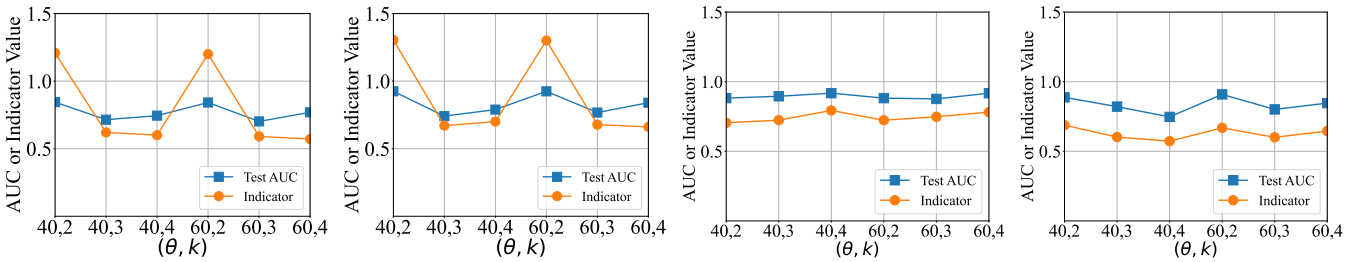
The performance gap between DPLP methods and LapGraph is influenced by the dataset's degree distribution. For instance, on the Yeast dataset, which has lowest average degree, LapGraph necessitates a high privacy budget (with $\varepsilon > 7.5$) to achieve satisfactory accuracy. Conversely, on PB, possessing the highest average degree, LapGraph surpasses the DPLP-NS baseline with a privacy budget of merely 2.5. Notably, the AUC of DPLP-PS method that of the non-private SEAL with much smaller privacy budgets and consistently outperforms LapGraph when $\varepsilon < 10$. This notable performance gap is attributed to LapGraph's direct perturbation of input, which alters the sparsity of the original graph. The sparsity of the graph serves as a crucial feature influencing the performance of link prediction models. Experimental results validate this observation, as LapGraph performs most poorly on the most

[3] http://vlado.fmf.uni-lj.si/pub/networks/data/bio/Yeast/Yeast.htm
[4] https://snap.stanford.edu/data/C-elegans-frontal.html



(a) C.ele (b) USAir (c) Yeast (d) PB

Fig. 5: Results of AUC by varying privacy budgets.



(a) C.ele (b) USAir (c) Yeast (d) PB

Fig. 6: Impact of parameter selection on the performance of DPLP-PS.

sparse Yeast dataset, with a AUC gap of over 30% compared to DPLP-PS when $\varepsilon < 2.5$.

2) Correctness of loss function design: We now validate the correctness of our designed loss function (see Equation 13) by studying its output parameters (θ, k) on the performance of DPLP. By varying combinations of (θ, k), we report the AUC of DPLP-PS on four datasets in Figure 6, where the privacy budget $\varepsilon = 11$ (with 0.1ε for parameter selection). Recall that in Equation 13, we capture the model loss $\ell(k, \theta)$ caused by projection and perturbation under given k and θ. Here we use reciprocal of the loss, i.e., $\frac{1}{\ell(k,\theta)}$, as an indicator, and show its values in Figure 6. We can observe that, the AUC and the indicator values show a similar trend across different combinations of (θ, k) on all datasets. This indicates that our designed loss function accurately captures the projection and perturbation loss, enabling DPLP-PS to benefit from parameter selection effectively.

On the USAir and PB datasets, fluctuations in data points are observed, and the prediction of trends for specific points deviates, such as the transition from point $(60, 3)$ to $(60, 4)$ on USAir. This is attributed to the relatively small size of these datasets, introducing biases in estimating projection and perturbation losses. However, even with such challenges, DPLP-PS remains viable to avoid unfavorable parameter selections on both datasets. This demonstrates the capability of our designed loss function to serve as a performance indicator and save privacy budget for expensive hyperparameter searching.

3) Ablation study on parameter selection: Now we further study the impact of parameters k and θ on the performance of DPLP-PS, respectively. As for the number of hops k, we restrict our selection of k to the set $\{2, 3, 4\}$. This decision is informed by two primary observations. Firstly, empirical assessments reveal that performance generally plateaus or does

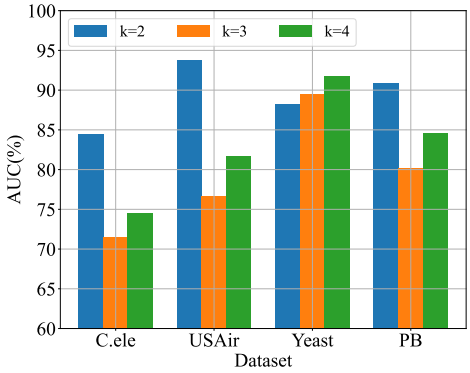
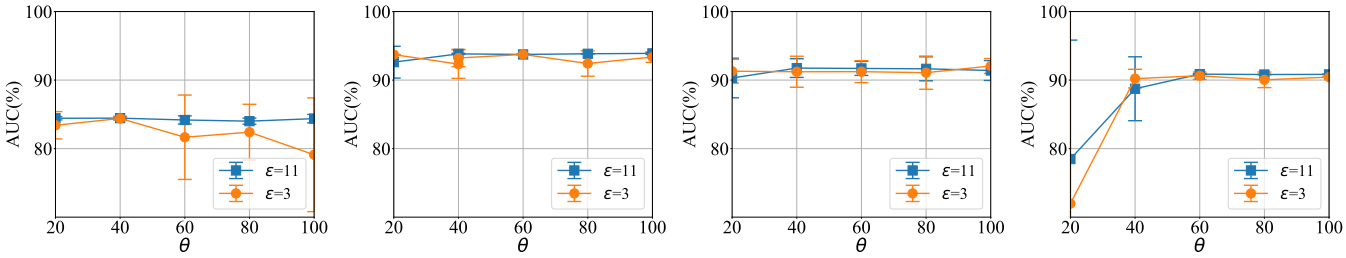


Fig. 7: Effect of the number of hops k on the AUC performance of the DPLP-PS on 4 datasets.

not improve significantly for $k \geq 5$ [6]. This aligns with our theoretical findings emphasizing that the most valuable information predominantly resides within shorter paths. Secondly, even at $k = 4$, the inclusion of a hub node can sometimes lead to exceedingly large subgraphs. Based on this setting, we report the AUC under a privacy budget of $\varepsilon = 4$. The findings are presented in Figure 7.

We observe that DPLP-PS exhibited peak performance for $k = 2$ on the C.ele, USAir, and PB datasets. For these datasets, as k increases to 4, the AUC of the DPLP-PS initially declines at $k = 3$ and then rebounds at $k = 4$. This fluctuation can be attributed to the fact that involving more hops augments the noise in the gradient, which in turn negatively impacts the model's AUC. Yet, with an increased k, the model can leverage information from more distant nodes for predictions, potentially counterbalancing the gradient perturbation. This phenomenon is particularly pronounced in datasets with a smaller average degree. As illustrated in Figure 7, DPLP-PS optimally capitalizes on multiple hops within the Yeast dataset,



(a) C.ele (b) USAir (c) Yeast (d) PB

Fig. 8: Impact of the threshold θ on the AUC performance of DPLP-PS.

achieving the highest efficacy at $k = 4$. This is because the information aggregated from more distant nodes is sufficient to compensate for the introduced noise from a larger k.

Then we study the influence of threshold θ on the AUC of DPLP-PS, where θ varies from 20 to 100 and the privacy budget ε is set to 3 or 11. Figure 8 shows that the AUC keeps growing with θ on PB (which has a high average degree), while on C.ele, USAir and Yeast (with lower average degrees), the AUC increases with θ up to a peak point around 40 or 60, and then becomes steady or decreases. This is due to the trade-off between having more samples for training and the amount of noise injected: the larger θ, the fewer samples are excluded from the aggregations (i.e., less information loss), but on the other hand, a larger sensitivity of the batched gradient introduces more noise.

4) Results with different GNN architectures: As mentioned in Section IV, the proposed DPLP paradigm can be implemented with most GNN architectures. We investigate the performance of DPLP-PS with three more GNN architectures apart from the default GCN architecture, namely GraphSAGE [32], GIN [33] and DGCNN [34]. The results are shown in the Table III.

We observed that GraphSAGE exhibits the least effective performance. This can be attributed to its method of employing the Hadamard product for pairwise node representations derived from a GNN, while omitting the use of a labeling trick [6]. Consequently, it fails to learn even basic neighborhood-overlap heuristics, confirming our arguments in the Introduction. Furthermore, our analysis shows a marginally better performance of GIN compared to GCN. This improvement stems from GIN's substitution of linear feature transformations in GCN with Multi-Layer Perceptrons (MLPs), enhancing its expressiveness in message-passing-based GNNs. As for DGCNN, it features a SortPooling readout layer that comprehensively processes all nodes in the enclosing subgraph. This methodology markedly improves its performance in specific datasets, notably Yeast, C.ele, and PB. However, there is a minor decline in effectiveness when applied to the USAir dataset. This variation in performance underlines the fact that employing a subgraph readout layer can be beneficial in certain contexts, a point that resonates with our initial discussion in the Introduction.

The aforementioned findings illustrate that DPLP is adaptable across a range of GNN architectures. Notably, even the GAE, which shows the least performance, can yield a model of acceptable usability. Despite the potential superiority of other models on varied datasets, GCN remains the preferred framework for DPLP. This preference is due to GCN's extensive application and its ability to outperform other architectures in certain scenarios, such as with the USAir dataset.

TABLE III: **Test AUC(%) of different GNN architectures with $\varepsilon = 4$. The best result on each dataset is highlighted.**

Model	Yeast	USAir	C.ele	PB
GCN	92.02 ± 1.18	$\mathbf{93.74 \pm 0.04}$	84.12 ± 0.29	90.75 ± 0.24
GraphSAGE	84.23 ± 1.25	86.29 ± 0.79	77.34 ± 2.37	80.53 ± 0.74
GIN	93.23 ± 1.24	93.48 ± 0.96	85.73 ± 0.18	91.79 ± 0.09
DGCNN	$\mathbf{94.76 \pm 0.96}$	93.33 ± 0.47	$\mathbf{88.75 \pm 0.54}$	$\mathbf{92.87 \pm 0.08}$

VIII. Conclusion

This work introduces a pioneering privacy-preserving link prediction framework, named DPLP, in the context of differential privacy. DPLP is the first work to learn differentially private GNNs specifically for link prediction, while being adaptable across various GNN architectures. Within DPLP framework, the subgraph extraction module serves as the building block to achieve high model utility. A neighborhood subgraph extraction method is first designed for DPLP, then we optimize it by integrating a novel path subgraph extraction scheme, which minimizes data dependency by concentrating on the information within the paths connecting the nodes of a target link. As a result, we have achieved notable improvements in prediction accuracy. Our extensive theoretical analysis, encompassing both privacy and utility dimensions, validates the framework's efficacy.

In future work, we aim to investigate whether the most appropriate subgraph construction method can be determined using our dataset knowledge. A key challenge involves devising a versatile sensitivity analysis method suitable for various subgraph construction techniques. Currently, we employ global sensitivity analysis, which accounts for the worst-case scenario. However, such scenarios rarely occur, particularly in graph data, where the hypothesized extreme cases might present a graph structure not representative of the original graph's actual structure. Our future objectives include tackling these challenges and extend the method to node privacy protection.

IX. Appendix

A. Proof of Lemma 3

Lemma (Link-level Sensitivity of Neighborhood Subgraph-based GNN.)**.** *Let G^θ be a graph with bounded degree θ, λ be the negative sampling rate, h be the radius of the neighborhood, $\mathcal{B}_t$ be the batch of subgraphs at any t step in Algorithm 2. Considering the batched gradient $\overline{\mathbf{g}}_t$ obtained from aggregating the gradient clipped by parameter C. Then the following inequality holds:*

$$\Delta(\overline{\mathbf{g}}_t) < 2C \cdot (1+\lambda)\left(\theta^{h+1} + \sum_{i=0}^{h-1} 2\theta^{i+1}\right) \tag{19}$$

Proof. Given two neighboring datasets G, G' differ on a link (s,q). Let $\Delta(\overline{\mathbf{g}}_t)$, $\Delta(\overline{\mathbf{g}'}_t)$ be the gradients computed on G and G', respectively. Let $\mathcal{T}_{\mathcal{B}_t} = \{(u,v) \mid (S_{uv}, y_{uv}) \in \mathcal{B}_t\}$ be the set of target links in $\mathcal{B}_t$ at iteration t trained on the subgraphs extracted from the θ degree bounded graph G^θ, and the negative sampling rate λ. The goal of sensitivity analysis is to compute the maximum l^2 norm difference between graident terms in a batch. Since the affected gradient terms are those derived on links in the contextual neighbors $LSN_h^g(s,q)$, we have the following inequality.

$$\begin{aligned}
&\|\overline{\mathbf{g}}_t - \overline{\mathbf{g}}'_t\|_F \\
&\leq \sum_{(u',v') \in LSN_h^g \cap \mathcal{T}_{\mathcal{B}_t}} \|\hat{\mathbf{g}}_t(S_{uv}, y_{uv}) - \hat{\mathbf{g}}'_t(S_{u'v'}, y_{u'v'})\|_F \\
&\leq \sum_{(u',v') \in LSN_h^g \cap \mathcal{T}_{\mathcal{B}_t}} \|\hat{\mathbf{g}}_t(S_{uv}, y_{uv})\| + \|\hat{\mathbf{g}}'_t(S_{u'v'}, y_{u'v'})\|_F \\
&\leq 2C \cdot M_g(h, \theta, \lambda)
\end{aligned}$$

By letting g be the function extracting neighborhood subgraph, we can substitue $M_g(h,\theta,\lambda)$ with its concrete equation, which completes the proof. □

B. Proof of Lemma 1

Lemma. *Consider the following link sets after the graph projection and negative sampling:*

- $e_1 = \{(u,v) \mid u \in \Gamma_h(s) \cap \Gamma_h(q), v \in N_1(u) \cup \bar{N}_1(u)\}$*;*
- $e_2 = \{(u,v) \mid u \in \Gamma_h(s) \backslash \Gamma_h(q), v \in \Gamma_h(q) \backslash \Gamma_h(s)\}$.

Then, $LSN_h^{g_n}(s,q) = e_1 \cup e_2$.

Proof. The contextual neighbors $LSN_h^{g_n}(s,q)$ comprise links whose h-hop neighborhood subgraph includes the pair (s, q). According to the definition, a link's neighborhood subgraph encompasses (s,q) if and only if $s, q \in \Gamma_h(u,v)$, where the following equality holds:

$$\begin{aligned}
&\Gamma_h(u,v) = \\
&(\Gamma_h(u) \setminus \Gamma_h(v)) \cup (\Gamma_h(v) \setminus \Gamma_h(u)) \cup (\Gamma_h(u) \cap \Gamma_h(v)).
\end{aligned}$$

Consequently, for any $(u,v) \in LSN_h^{g_n}(s,q)$, there exist two scenarios: In the first, both s and q are in the neighborhood of either u or v, denoted as $s, q \in \Gamma_h(u)$ or $\Gamma_h(v)$. In this scenario, any target link, including negative links incident to u or v, may be a subgraph contextual neighbor of (s,q), forming the set $e_1 = \{(u,v) \mid u \in \Gamma_h(s) \cap \Gamma_h(q), v \in N_1(u) \cup \bar{N}_1(u)\}$. In the second scenario, s and q belong to the neighborhoods of two distinct nodes, specifically, $s \in \Gamma_h(u) \setminus \Gamma_h(v)$ and $q \in \Gamma_h(v) \setminus \Gamma_h(u)$. Accordingly, the positions of (u,v) can be defined as $u \in \Gamma_h(s) \setminus \Gamma_h(q)$ and $v \in \Gamma_h(q) \setminus \Gamma_h(s)$, constituting the link set e_2. As the graph is undirected, link directionality, specifically the reversal of u and v, is not considered. The proof is compelete. □

C. The effect of learning rate and batch size

Based on the results of parameter selection, we can ascertain the remaining optimal hyperparameters including batch size, learning and the maximum number of epochs of DPLP-PS for each dataset as shown in the following figures 9 to 11.

References

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks. journal of the association for information science and technology (2007)," *Google Scholar Google Scholar Digital Library Digital Library*, 2007.

[2] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[3] J. Bennett, S. Lanning *et al.*, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007. New York, 2007, p. 35.

[4] N. Shibata, Y. Kajikawa, and I. Sakata, "Link prediction in citation networks," *Journal of the American society for information science and technology*, vol. 63, no. 1, pp. 78–85, 2012.

[5] M. Zhang, P. Li, Y. Xia, K. Wang, and L. Jin, "Revisiting graph neural networks for link prediction," 2020.

[6] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[7] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[8] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[9] F. Wu, Y. Long, C. Zhang, and B. Li, "Linkteller: Recovering private edges from graph neural networks via influence analysis," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2005–2024.

[10] S. Sajadmanesh, A. S. Shamsabadi, A. Bellet, and D. Gatica-Perez, "Gap: Differentially private graph neural networks with aggregation perturbation," in *USENIX Security 2023-32nd USENIX Security Symposium*, 2023.

[11] S. Sajadmanesh and D. Gatica-Perez, "Locally private graph neural networks," in *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, 2021, pp. 2130–2145.

[12] W. Lin, B. Li, and C. Wang, "Towards private learning on decentralized graphs with local differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2936–2946, 2022.

[13] A. Daigavane, G. Madan, A. Sinha, A. G. Thakurta, G. Aggarwal, and P. Jain, "Node-level differentially private graph neural networks," *arXiv preprint arXiv:2111.15521*, 2021.

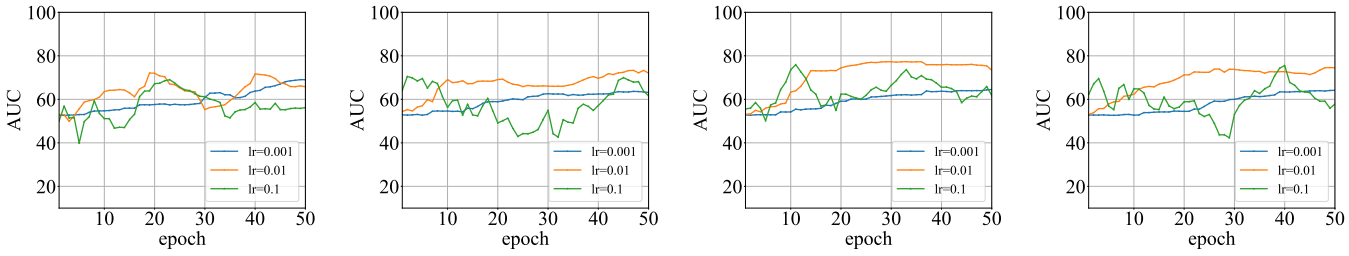
[14] X. Ran, "Differentially private graph neural networks for link prediction," https://github.com/marlowe518/dplp/blob/master/Main.pdf, 2023.

[15] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

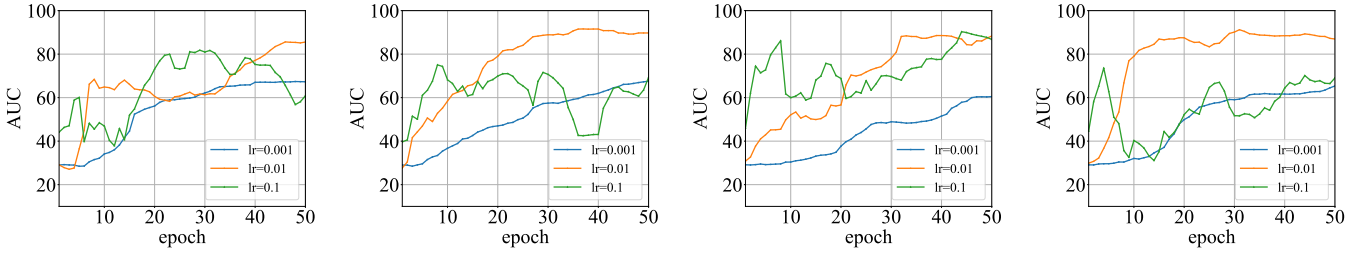
[17] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," *Recommender systems handbook*, pp. 73–105, 2011.

[18] U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft, "Blurme: Inferring and obfuscating user gender based on ratings," in *Proceedings of the sixth ACM conference on Recommender systems*, 2012, pp. 195–202.



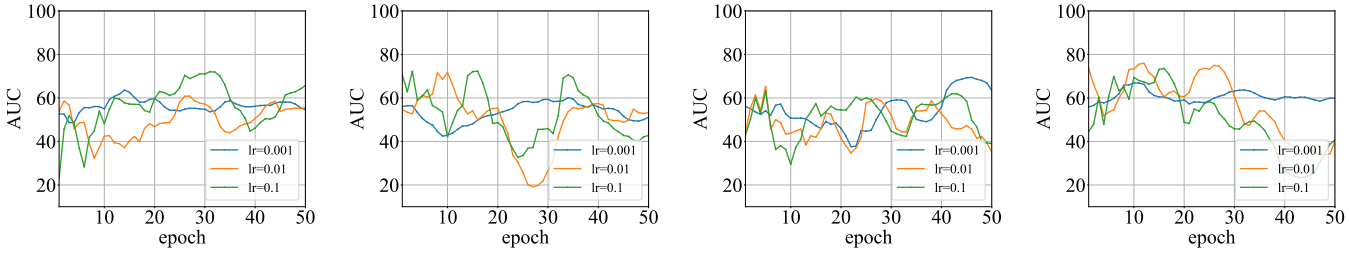
(a) **(b)** **(c)** **(d)**

Fig. 9: The AUC trend on validation set of DPLP-PS on C.ele: (a): batch size=32; (b): batch size=128; (c): batch size=1024; (d): batch size=4096 (P.S. The k and θ are set according to the best performance in the following experiments)



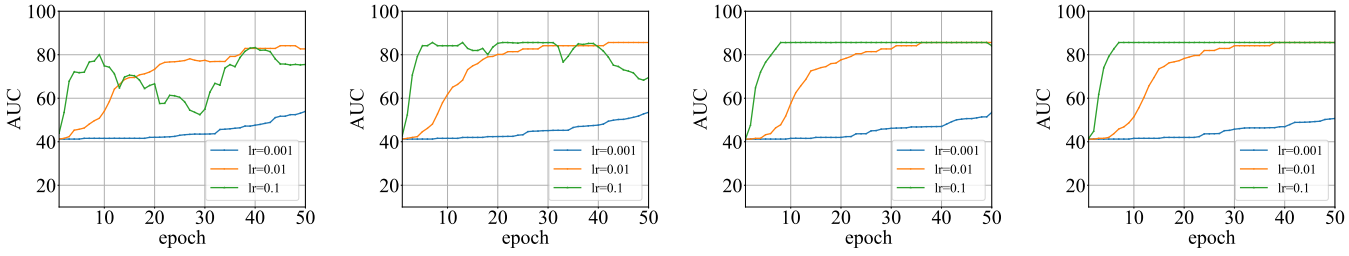
(a) **(b)** **(c)** **(d)**

Fig. 10: The AUC trend on validation set of DPLP-PS on USAir: (a): batch size=32; (b): batch size=128; (c): batch size=1024; (d): batch size=4096 (P.S. The k and θ are set according to the best performance in the following experiments)



(a) **(b)** **(c)** **(d)**

Fig. 11: The AUC trend on validation set of DPLP-PS on Yeast: (a): batch size=32; (b): batch size=128; (c): batch size=1024; (d): batch size=4096 (P.S. The k and θ are set according to the best performance in the following experiments)



(a) **(b)** **(c)** **(d)**

Fig. 12: The AUC trend on the validation set of DPLP-PS on PB: (a): batch size=32; (b): batch size=128; (c): batch size=1024; (d): batch size=4096 (P.S. The k and θ are set according to the best performance in the following experiments)

[19] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 111–125.

[20] A. Epasto, V. Mirrokni, B. Perozzi, A. Tsitsulin, and P. Zhong, "Differentially private graph learning via sensitivity-bounded personalized pagerank," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 617–22 627, 2022.

[21] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov, "" you might also like:" privacy risks of collaborative filtering," in *2011 IEEE symposium on security and privacy*. IEEE, 2011, pp. 231–246.

[22] A. De and S. Chakrabarti, "Differentially private link prediction with protected connections," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 1, 2021, pp. 63–71.

[23] S. P. Liew, T. Takahashi, S. Takagi, F. Kato, Y. Cao, and M. Yoshikawa, "Network shuffling: Privacy amplification via random walks," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 773–787.

[24] A. Friedman, S. Berkovsky, and M. A. Kaafar, "A differential privacy framework for matrix factorization recommender systems," *User Modeling and User-Adapted Interaction*, vol. 26, pp. 425–458, 2016.

[25] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep Learning with Differential Privacy," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, Oct. 2016.

[26] T. T. Mueller, J. C. Paetzold, C. Prabhakar, D. Usynin, D. Rueckert, and G. Kaissis, "Differentially private graph classification with gnns," *arXiv preprint arXiv:2202.02575*, 2022.

[27] H. Sun, X. Xiao, I. Khalil, Y. Yang, Z. Qin, H. Wang, and T. Yu, "Analyzing subgraph statistics from extended local views with decentralized differential privacy," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 703–717.

[28] Y.-X. Wang, B. Balle, and S. P. Kasiviswanathan, "Subsampled rényi differential privacy and analytical moments accountant," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1226–1235.

[29] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 2017, pp. 263–275.

[30] A. Kumar, S. S. Singh, K. Singh, and B. Biswas, "Link prediction techniques, applications, and performance: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 553, p. 124289, 2020.

[31] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[32] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[33] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[34] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An End-to-End Deep Learning Architecture for Graph Classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.